WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Monday, January 26, 2004

Hide?	Set Name	Query	Hit Count
	DB=USPT; PLUR=YES; OP=ADJ		
□	L28	L27 and l18	3
□	L27	l22 same (self adj (describ$ or contain$))	22
□	L26	L25 and l22	2
□	L25	(rpc or (remot$ adj procedur$ adj call$)) same (self adj (describ$ or contain$))	7
□	L24	l23 same (self adj (describ$ or contain$))	1
□	L23	l22 same (rpc or (remot$ adj procedur$ adj call$))	113
□	L22	dcom or (distribut$ com) or (distribut$ adj component adj object adj model$)	476
□	L21	(object$ near4 (self adj (describ$ or contain$))) same (rpc or (remot$ adj procedur$ adj call$))	2
□	L20	5764915[pn]	1
□	L19	l13 and l18	4
□	L18	719/311-332[ccls]	2286
□	L17	l13 and L16	0
□	L16	719/318[ccls]	190
□	L15	(object$ near4 (self adj (describ$ or contain$))) same ((defer$ or delay$ or laten$) near4 (rebuild$ or reconstruct$ or build$ or construct$))	2
□	L14	L13 and l4	2
□	L13	(object$ near4 (self adj (describ$ or contain$))) near12 (rebuild$ or reconstruct$ or build$ or construct$)	190
	DB=EPAB,DWPI; PLUR=YES; OP=ADJ		
□	L12	(object$ near4 (self adj (describ$ or contain$))) near12 (rebuild$ or reconstruct$ or build$ or construct$)	2
□	L11	(rpc or (remot$ adj procedur$ adj call$)) and ((delay$ or defer$ or laten$) near4 (rebuild$ or reconstruct$ or build$ or construct$))	1
□	L10	(rpc or (remot$ adj procedur$ adj call$)) same (object$ near4 (rebuild$ or reconstruct$ or build$ or construct$))	2
	DB=PGPB; PLUR=YES; OP=ADJ		
□	L9	(rpc or (remot$ adj procedur$ adj call$)) same (object$ near4 (rebuild$ or reconstruct$ or build$ or construct$))	13
□	L8	719/330[ccls]	85
□	L7	(rpc or (remot$ adj procedur$ adj call$)) same ((delay$ or defer$ or laten$) near4 (rebuild$ or reconstruct$ or build$ or construct$))	2
	DB=USPT; PLUR=YES; OP=ADJ		

US006212578B1

(12) United States Patent

Racicot et al.

(10) **Patent No.: US 6,212,578 B1**

(45) **Date of Patent: Apr. 3, 2001**

(54) **METHOD AND APPARATUS FOR MANAGING DEPENDENCIES IN A DISTRIBUTED COMPUTING ENVIRONMENT FOR ENSURING THE SAFETY OF REMOTE PROCEDURE CALLS**

(75) Inventors: **Christopher Racicot**, Mountain View; **Usha Sundaram**, Sunnyvale, both of CA (US)

(73) Assignee: **Oracle Corporation**, Redwood Shores, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **08/711,772**

(22) Filed: **Sep. 9, 1996**

(51) **Int. Cl.**^{7} **G06F 1/12**

(52) **U.S. Cl.** **709/330**

(58) **Field of Search** 395/684; 709/300–305; 713/100

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,475,819	*	12/1995	Miller et al.	395/200.03
5,511,197	*	4/1996	Hill et al.	395/683
5,671,414	*	9/1997	Nicolet	395/684
5,682,534	*	10/1997	Kapoor et al.	395/684
5,778,228	*	7/1998	Wei	395/684
5,887,172	*	3/1999	Vasudevan et al.	709/304

OTHER PUBLICATIONS

Burke et al., "RPC Design for Real–Time Mach", Open Software Foundation/Research Institute, pp. 1–35, Apr. 12, 1994.*

Bloomer, John, "Distributed Computing and the OSF/DCE", Dr. Dobb's Journal, (33), Feb. 1995.*

Shapiro, et al., "Remote Procedure Calls", published on WWW:http://bumetb.bu.edu/~bruce/cs776/projects/fall97/shapiro/shapiro.htm, Dec. 1997.*

(No author given), "COBRA Security", published by OMG, pp. 80, 303, Dec. 1995.*

* cited by examiner

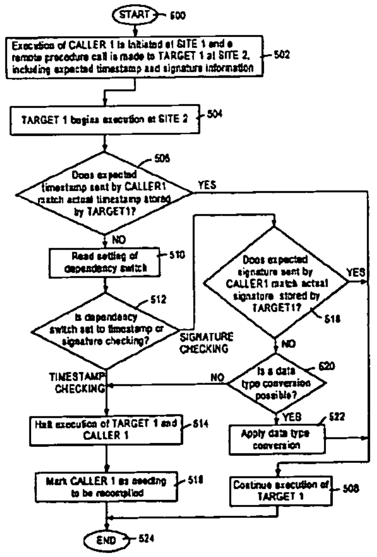
Primary Examiner—Alvin E. Oberley
Assistant Examiner—St. John Courtenay, III
(74) *Attorney, Agent, or Firm*—Hickman Palermo Truong & Becker, LLP; Edward A. Becker

(57) **ABSTRACT**

A method and apparatus for managing dependencies in a distributed environment to ensure the safety of remote procedure calls is disclosed. Each remote procedure call between a calling procedure and a target procedure includes an expected time stamp and expected signature associated with the target procedure. The expected time stamp contains the creation time of the target procedure at the time the calling procedure was last compiled. The expected signature contains data type information of the target procedure formal parameters at the time the calling procedure was last compiled.

The target procedure compares the expected time stamp to an actual time stamp maintained by the target procedure. If the two time stamps do not match, the target procedure compares the expected signature and an actual signature maintained by the target procedure to determine whether the data types of the formal parameters sent by the calling procedure are compatible with the data types expected by the target procedure. If the data types are compatible, then execution of the target procedure continues.

41 Claims, 6 Drawing Sheets


START
500
Execution of CALLER 1 is initiated at SITE 1 and a remote procedure call is made to TARGET 1 at SITE 2, including expected timestamp and signature information
502
TARGET 1 begins execution at SITE 2
504
Does expected timestamp sent by CALLER1 match actual timestamp stored by TARGET1?
506
YES
NO
Read setting of dependency switch
510
Is dependency switch set to timestamp or signature checking?
512
SIGNATURE CHECKING
TIMESTAMP CHECKING
Does expected signature sent by CALLER1 match actual signature stored by TARGET1?
518
YES
NO
Is a data type conversion possible?
520
NO
YES
Apply data type conversion
522
Halt execution of TARGET 1 and CALLER 1
514
Mark CALLER 1 as needing to be recompiled
516
Continue execution of TARGET 1
508
END
524

First Hit Fwd Refs

□ Generate Collection Print

L25: Entry 4 of 7 File: USPT Apr 3, 2001

DOCUMENT-IDENTIFIER: US 6212578 B1
TITLE: Method and apparatus for managing dependencies in a distributed computing environment for ensuring the safety of remote procedure calls

Brief Summary Text (7):
Historically, several approaches have been used to manage dependencies in distributed computing environments to ensure compatibility during remote procedure calls. Four of these include (1) synchronized installation; (2) time stamps; (3) self-describing data; and (4) data type encoding.

Brief Summary Text (13):
Perhaps the most widely used approach for ensuring the safety of remote procedure calls in distributing computing environments is the use of self-describing data. With this approach, additional data is included in each remote procedure call which fully describes each parameter. This data typically includes type, mode, constraints, and any other meta-data required to fully describe the parameter and ensure correctness.

Brief Summary Text (15):
However, the self-describing data approach adversely affects performance in two ways. First, the self-describing data greatly increases the amount of data being passed in each remote procedure call. Secondly, the data type information is typically interleaved with the parameters, requiring that all of the parameters be checked before compatibility can be confirmed. Consequently, a difference in the last parameter will not be detected until all of the other parameters have been checked.

US006651099B1

(12) **United States Patent**
Dietz et al.

(10) **Patent No.: US 6,651,099 B1**
(45) **Date of Patent: Nov. 18, 2003**

(54) **METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK**

(75) Inventors: **Russell S. Dietz**, San Jose, CA (US); **Joseph R. Maixner**, Aptos, CA (US); **Andrew A. Koppenhaver**, Littleton, CO (US); **William H. Bares**, Germantown, TN (US); **Haig A. Sarkissian**, San Antonio, TX (US); **James F. Torgerson**, Andover, MN (US)

(73) Assignee: **Hi/fn, Inc.**, Los Gatos, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 589 days.

(21) Appl. No.: **09/608,237**

(22) Filed: **Jun. 30, 2000**

Related U.S. Application Data bad

(60) Provisional application No. 60/141,903, filed on Jun. 30, 1999.

(51) **Int. Cl.**^{7} **G06F 13/00**

(52) **U.S. Cl.** **709/224**; 370/389

(58) **Field of Search** 709/200, 201, 709/220, 223, 224, 231, 232, 236, 238, 239, 240, 246; 370/389, 392, 395.32

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,736,320 A	4/1988	Bristol	364/300
4,891,639 A	1/1990	Nakamura	340/825.5
5,101,402 A	3/1992	Chui et al.	370/17
5,247,517 A	9/1993	Ross et al.	370/85.5
5,247,693 A	9/1993	Bristol	395/800
5,249,292 A	9/1993	Chiappa	395/650
5,315,580 A	5/1994	Phaal	370/13
5,339,268 A	8/1994	Machida	365/49
5,351,243 A	9/1994	Kalkunte et al.	370/92
5,365,514 A	11/1994	Hershey et al.	370/17
5,375,070 A	12/1994	Hershey et al.	364/550
5,394,394 A	2/1995	Crowther et al.	370/60

(List continued on next page.)

OTHER PUBLICATIONS

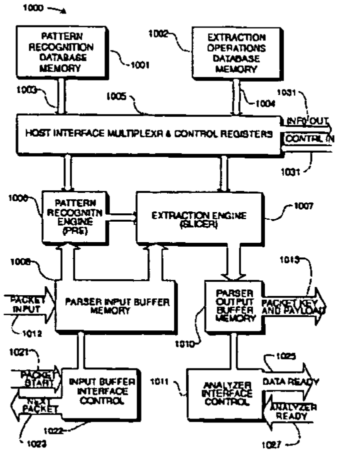
"Technical Note: the Narus System," Downloaded Apr. 29, 1999 from www.narus.com, Narus Corporation, Redwood City California.

Primary Examiner—Moustafa M. Meky
(74) *Attorney, Agent, or Firm*—Dov Rosenfeld; Inventek

(57) **ABSTRACT**

A monitor for and a method of examining packets passing through a connection point on a computer network. Each packets conforms to one or more protocols. The method includes receiving a packet from a packet acquisition device and performing one or more parsing/extraction operations on the packet to create a parser record comprising a function of selected portions of the packet. The parsing/extraction operations depend on one or more of the protocols to which the packet conforms. The method further includes looking up a flow-entry database containing flow-entries for previously encountered conversational flows. The lookup uses the selected packet portions and determining if the packet is of an existing flow. If the packet is of an existing flow, the method classifies the packet as belonging to the found existing flow, and if the packet is of a new flow, the method stores a new flow-entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry. For the packet of an existing flow, the method updates the flow-entry of the existing flow. Such updating may include storing one or more statistical measures. Any stage of a flow, state is maintained, and the method performs any state processing for an identified state to further the process of identifying the flow. The method thus examines each and every packet passing through the connection point in real time until the application program associated with the conversational flow is determined.

10 Claims, 18 Drawing Sheets


1000
PATTERN RECOGNITION DATABASE MEMORY
1001
1002
EXTRACTION OPERATIONS DATABASE MEMORY
1003
1005
1004
1031
INFO OUT
HOST INTERFACE MULTIPLEXR & CONTROL REGISTERS
CONTRL IN
1031
1006
PATTERN RECOGNITN ENGINE (PRE)
EXTRACTION ENGINE (SLICER)
1007
1008
1013
PACKET INPUT
PARSER INPUT BUFFER MEMORY
PARSER OUTPUT BUFFER MEMORY
PACKET KEY AND PAYLOAD
1012
1010
1021
1025
PACKET START
INPUT BUFFER INTERFACE CONTROL
1011
ANALYZER INTERFACE CONTROL
DATA READY
NEXT PACKET
ANALYZER READY
1022
1023
1027

D Com

RPC's that allow self-contained objects

First Hit Fwd Refs

□ Generate Collection Print

L21: Entry 1 of 2 File: USPT Nov 18, 2003

DOCUMENT-IDENTIFIER: US 6651099 B1
TITLE: Method and apparatus for monitoring traffic in a network

Brief Summary Text (12):
Other protocols that may lead to disjointed flows, include RPC (Remote Procedure Call); DCOM (Distributed Component Object Model), formerly called Network OLE (Microsoft Corporation, Redmond, Wash.); and CORBA (Common Object Request Broker Architecture). RPC is a programming interface from Sun Microsystems (Palo Alto, Calif.) that allows one program to use the services of another program in a lo remote machine. DCOM, Microsoft's counterpart to CORBA, defines the remote procedure call that allows those objects--objects are self-contained software modules--to be run remotely over the network. And CORBA, a standard from the Object Management Group (OMG) for communicating between distributed objects, provides a way to execute programs (objects) written in different programming languages running on different platforms regardless of where they reside in a network.